1/5/1 (Item 1 from file: 351)
DIALOG(R)File 351:Derwent WPI

010777139 **Image available**
WPI Acc No: 1996-274092/ **199628**
XRPX Acc No: N96-230484

Control program generation system of communication program counter - controls incorrect operation in order of generation of event using event buffer

Patent Assignee: HITACHI LTD (HITA)
Number of Countries: 001 Number of Patents: 001
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
JP 8115107	A	19960507	JP 94250350	A	19941017	199628	B

Priority Applications (No Type Date): JP 94250350 A 19941017
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
JP 8115107	A		6	G05B-015/02	

Abstract (Basic): JP 8115107 A

The system stores the event in an event buffer temporarily, when its transition state is not defined by processing during reading operation of its algorithm. Then state transition of the event is performed.

The state transition diagram of the event is explained by a table using a graphical user interface. According to an algorithm a state transition model is updated. The incorrect operation and the order of event generation is controlled by the event buffer.

ADVANTAGE - Produces errorless transition state table in desired range. Improves reliability of system. Prevents dead lock and incorrect operation of system. Improves production efficiency of software.

Dwg.1/8

Title Terms: CONTROL; PROGRAM; GENERATE; SYSTEM; COMMUNICATE; PROGRAM; COUNTER; CONTROL; INCORRECT; OPERATE; ORDER; GENERATE; EVENT; EVENT; BUFFER
Derwent Class: T01; T06
International Patent Class (Main): G05B-015/02
International Patent Class (Additional): G06F-009/06; G06F-011/34
File Segment: EPI

1/5/2 (Item 1 from file: 347)
DIALOG(R)File 347:JAPIO

05159607 **Image available**
EQUIPMENT CONTROL PROGRAM GENERATING SYSTEM

PUB. NO.: 08-115107 [**JP 8115107** A]
PUBLISHED: May 07, 1996 (19960507)
INVENTOR(s): SUZUKI HIROYUKI
APPLICANT(s): HITACHI LTD [000510] (A Japanese Company or Corporation), JP (Japan)
APPL. NO.: 06-250350 [JP 94250350]
FILED: October 17, 1994 (19941017)
INTL CLASS: [6] G05B-015/02; G06F-009/06; G06F-011/34
JAPIO CLASS: 22.3 (MACHINERY -- Control & Regulation); 45.1 (INFORMATION PROCESSING -- Arithmetic Sequence Units)

ABSTRACT

PURPOSE: To prevent malfunction or deadlock from occuring by providing an event storage buffer so that the malfunction depending on the generation order of events can not occur.

CONSTITUTION: When any state transition is not defined, an event is stored in the event buffer and processing is returned to event waiting (105). When the state transition is defined, any action is executed and state updating is executed (115 and 120). The new state transited by the action and updating is defined at a position, where the original state crosses the read event, in a state transition table. Corresponding to the transition to the new state, it is investigated whether the state transition occurs because of the event in the event buffer or not. Besides, the event is extracted from the event buffer (125), it is decided whether the state transition occurs corresponding to the new state or not and when the state transition is not defined, the next event is extracted. When it is defined, processing is branched to the action execution (115).

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平8－115107

(43)公開日　平成８年(1996)５月７日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０５Ｂ　15/02				
Ｇ０６Ｆ　9/06	５３０　Ａ	7230－5B		
11/34	Ｌ	7313－5B		C3
		7740－3H	Ｇ０５Ｂ　15/ 02	Ｐ

審査請求　未請求　請求項の数３　ＯＬ　（全６頁）

(21)出願番号　特願平6－250350

(22)出願日　平成６年(1994)10月17日

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地

(72)発明者　鈴木　博之
茨城県勝田市大字市毛882番地　株式会社日立製作所計測器事業部内

(74)代理人　弁理士　小川　勝男

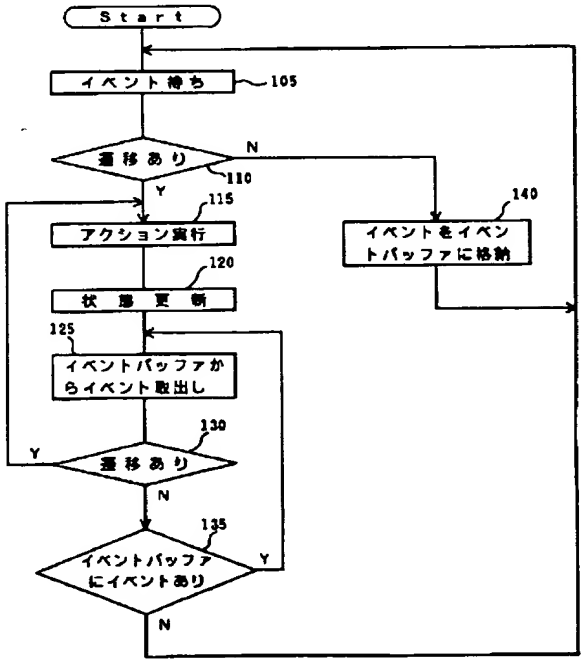
(54)【発明の名称】　機器制御プログラム生成方式

(57)【要約】

【構成】状態遷移表によって定義された状態遷移マシンにおいて、イベントバッファを設け、アルゴリズムによって処理を行うことによって状態遷移の定義されていないイベントが読み込まれた場合に一時的にイベントバッファに格納し、他の状態に遷移した時に取り出して状態遷移を行う。また状態遷移図を表示し、グラフィカルユーザインターフェースを通して状態遷移表の内容を表示し、アルゴリズムに従って状態遷移モデルを更新し、実際の状態遷移マシンで解釈できる状態遷移表を得る。

【効果】定義されていないイベントは状態マシンが別状態に移るまで処理されずにイベントバッファに格納され、定義されたイベントがくるまで待つのでプログラムの信頼性は向上し、また設計時の制約条件が緩和される。

図　1



1

【特許請求の範囲】

【請求項１】ハードウェアを操作するためのソフトウェア・インターフェース，マルチプログラミングをサポートするためのタスク管理機能とタスク間通信機能を持ったオペレーティングシステム，イベント発生機構および読取り機構，制御情報を記述した状態遷移表からなる機器制御プログラムにおいて、イベント記憶バッファを具備することによってイベントの発生順序に依存した誤動作をしないことを特徴とする機器制御プログラム生成方式。

【請求項２】請求項１において、状態遷移図を表示するディスプレイと，表示されている状態遷移図の部分を指し示すポインティングデバイスを有し、ポインティングデバイスによって示された、ある状態から他の状態へのパスのみが有効な状態遷移として認識され、それ以外のイベントの発生を別状態に移るまで処理を遅らせる機器制御プログラム生成方式。

【請求項３】請求項１または２において、前記機器制御プログラムに対して、状態遷移およびイベントの発生，処理時刻を記録するデータベースを具備することによって、イベントの待ち時間をモニタする機器制御プログラム生成方式。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は通信プログラムや機器制御プログラムの作成方式に関する。

【０００２】

【従来の技術】機器制御プログラムの作成において、状態モデルが使われることが多いが、状態の数が多いとモデルの正当性，完全性を検証することが難しかった。特に状態遷移モデルを作成する上での難易性は、全ての状態においてあり得る全てのイベント入力に対して状態遷移を定義する必要があり、状態遷移の洩れがあるとシステムのデッドロックや誤動作となってしまった。このため、状態数の多いシステムでは状態管理を統一的な手法で行わず、共有メモリ，セマフォーなどによってシステムの状態を監視しながら処理を行うのが普通であった。

【０００３】これらの状態遷移モデルを設計／作図する作業はグラフィカルな状態遷移図を作成し、状態遷移洩れのないように状態遷移表を注意深く埋め、プログラムに書き込んでいた。これらの工程は従来人手によって行われていたために、設計からコーディングまでの間に入力ミスが入る可能性があった。また通常、状態遷移モデルの設計時においてシステムの動的な振舞いを設計するために状態遷移図を用い、イベントに対する状態遷移の定義洩れや誤りをなくし、個々の状態に対して全てのイベントに対する状態遷移を定義するために状態遷移表が用いられていた。

【０００４】一般的に状態遷移モデルは、時間の概念を明確に表現することができないため、正常動作を確認しても、それが最適な動作か否かを知ることができなかった。状態モニタはシステムの実行時に個々のサブシステムの動作の様子をタイムチャートに示してあるイベントを待っている無駄時間の調査を支援するためのものである。

2

【０００５】

【発明が解決しようとする課題】機器制御プログラムはハードウェアとの関連が強いため、汎用的なアルゴリズムを利用する試みがなされていなかった。一般に機器制御のアルゴリズムとして状態遷移方式や決定表方式が使われているが、システムの規模が大きくなるとシステムの状態数が急激に増えてしまい、複雑になり過ぎてしまう。この結果、状態モデルの検証が難しくなり、状態遷移表に状態の定義洩れ，状態遷移条件や遷移先の誤りや定義洩れなどの設計不良が入り込みやすくなり、またそれの検出は難しくなる。

【０００６】本発明の目的は、複雑なシステムに対して状態遷移洩れのない完全な状態モデルを構築しなくても、操作の基本的なサイクルさえ定義されていれば誤動作やデッドロックを引き起こすことなく、正常に要求された機能を実行する制御方式を提供することにある。

【０００７】状態モデルを作成するために、従来併用されていた状態遷移表／状態遷移図をより直観的でシステムの動作を理解しやすい状態遷移図のみで行い、状態遷移表を使った遷移洩れのチェック作業をなくし、モデルを単純化し、信頼性，プログラムの生産性を向上させることである。

【０００８】一般的に状態遷移モデルは、時間の概念を明らかに表現することができないため、正常動作を確認しても、それが最適な動作か否かを知ることができない。

【０００９】本発明の目的は個々のサブシステムの動作の様子をタイムチャートなどの形式でグラフィカルに表示することによって、イベントを待っている無駄時間の調査を支援することにある。

【００１０】

【課題を解決するための手段】状態遷移図中には、状態遷移のみを引き起こし、装置に対して何のアクションも行わないイベントが存在する。たとえばある状態においてイベントＥ１が起こると状態Ｓ１に遷移してアクションＰ１を実行し、ついでイベントＥ２が起こると状態Ｓ２に遷移するが、イベントの発生がＥ２，Ｅ１の順番だとダミーの状態Ｓ３に遷移して何のアクションも行わず、次のＥ１のイベントを受けて状態Ｓ４に遷移してＰ１とＰ２を同時に処理する場合である。

【００１１】グラフィックディスプレイを利用し、状態遷移図を表示したり、状態遷移図の有効グラフをポインティングデバイスなどのユーザインターフェースから編集することによって状態モデルを作成する。またプログラムで解釈される状態遷移表データとグラフィックディ

スプレイ上の状態遷移図との対応づけを維持することによって状態遷移表データの設計を状態遷移図の作画作業に置き替える。

【００１２】各々の状態機械の現状態を示す状態変数を他のプログラムから参照できる領域に確保し、またシステム内で発生した全てのイベントを監視し、状態機械の状態が遷移した時点を常に捉え、その状態遷移に対してグラフィカル端末制御プログラムに、状態機械の状態遷移を表示する命令を発行することによって、常にシステムにおける状態機械の状態を把握することができる。また、状態遷移の時間レンジが対話的に監視するには極端に短かったり、あるいは長い場合には状態遷移を表示する命令を発行時刻を追加して、ロギングし、システムの制御後、オフライン的にそのログ情報の一部または全部をグラフィック端末上にタイムチャート的に表示することによって目的の無駄時間調査を行うことができる。

【００１３】

【作用】イベントバッファは従来の状態遷移マシンではデッドロックしてしまう、定義洩れのある状態遷移モデルに対してもデッドロックなどの危険性を減少する。また実際に動かしたい状態遷移のパスだけを正確に定義すれば、イベントの発生順序が乱れても定義通りに処理を実行することができる。

【００１４】状態遷移表は状態マシンの動きを記述するためにはコンパクトな記述方法であるが、直観的な表現ではない。ＧＵＩの利用仕方によってさらにプログラムの開発が容易となり、設計洩れ，入力ミスなどの危険性が減少する。

【００１５】全ての状態において、全てのイベントに対する状態遷移が定義されているとは限らず、その結果、直観を元にして作成した状態遷移モデルはイベントが発生されてから実際に処理されるまでに長い間、イベントバッファに格納されたままになり実行効率が低い制御システムとなる可能性がある。状態遷移マシンにおける状態とイベントバッファに格納されたイベントの様子を時系列的に表示することにより、性能上の問題点を容易に見つけ出し、状態モデルの改良に関する洞察を得ることができる。

【００１６】

【実施例】イベントバッファを用いた状態遷移マシンのプログラムフローチャートを図１に、状態遷移表の構造を図２に、イベントバッファの構造を図３に示す。本実施例では状態遷移モデルとしてMealy モデルについて述べる。

【００１７】図１において状態遷移プログラムは初期状態Ｓ０が与えられて起動され、イベント待ち１０５となる。状態遷移プログラムにイベントが送られると図２の状態遷移表を参照し、状態Ｓ０（２１０）に対してイベント２０５による状態遷移が定義されているかを調べ、遷移ありなしの判定１１０を行う。もし、状態遷移が定

義されていなければ、イベントを図３に示すイベントバッファに格納してイベント待ち１０５に戻る。イベントバッファは管理部として最も新しく追加されたイベントデータを指す最新ポインタ３０５と最も前に登録された先頭ポインタ３１０の情報を持つ。イベントバッファに格納されているイベント３１５はイベントコード３２０を持ち、どの要因によってイベントが発生したかが判別できる。新しいイベントが追加される場合は最新ポインタ３０５が指すイベントのリンクポインタに新しいイベントのアドレスを入れ、新しいイベントのリンクポインタには終端記号としてＮＵＬＬを代入する。

【００１８】１１０にて読み込んだイベントに対して状態遷移が定義されている場合はアクション実行１１５，状態更新１２０を実行する。アクションと更新によって遷移する新しい状態は状態遷移表の中で、元の状態と読み込んだイベントの交わったところに定義されている。

【００１９】新しい状態に遷移したことによって、イベントバッファ内のイベントによって状態遷移が起こるかを調べる。１２５でイベントバッファからイベントを取り出し、新しい状態に対して状態遷移を起こすか否かを判定し（１３０）、状態遷移が定義されていなければイベントバッファから次のイベントを取り出し、同様に状態遷移の定義を調べる。状態遷移が定義されていればアクション実行１１５に分岐する。イベントバッファの全てのイベントに対して状態遷移が定義されていなければイベント待ち１０５に分岐して、以上の処理を繰り返す。なお、ここでイベントバッファからイベントを取り出し１２５は、イベントのポインタを取り出すだけであって、実際にイベントバッファのリンクド・リストから取り除くことではない。

【００２０】上記のような状態遷移マシンを生成するための手段として、グラフィカルユーザインターフェースを用いた実施例を図４および図５に示す。図４はイベント記憶を用いた状態遷移マシン生成のためのグラフィカルユーザインターフェースのための画面４００である。画面４００で表示されている情報は状態ラベルのついた状態情報４０５，状態から状態への遷移４１０，状態遷移属性を定義するためのウィンドウ４１５，新しい状態を生成するための生成アイコン４２０，状態マシン生成を終了するための終了アイコン４２５である。

【００２１】上記の画面４００から実際にプログラムを生成するためのアルゴリズムが図５に示されている。最初にＧＵＩ（グラフィカルユーザインターフェース）からのイベント待ち５０５状態に入り、終了アイコンが押されたらプログラムを終了する。生成アイコンが押されたら新しい状態を画面上に作成し、位置決めを行って新状態作図５２０を実行し、状態遷移表に新状態を追加する（５３０）。画面上の状態情報を直接ポインティングデバイスで選択されたら状態遷移作図５１５を実行し、その中で、遷移先状態をポインティングデバイスで指定

5

し、両状態の間に方向を示す情報を持ったリンクを作図する。続いて属性定義ウィンドウ表示５２５を行って、リンクのための属性入力をユーザに促し、イベント属性読み取り５３５を行う。イベント属性として、イベントコード，アクション，イベントデータなどを定義する。最後に状態遷移表に状態遷移を追加５４０し、ＧＵＩイベント待ち５０５に戻る。

【００２２】上記の状態遷移マシンは必ずしも、イベントが発生したら最短時間でアクションを実行する最適なシステムとは限らないため、イベントが発生してから実際に処理されるまでの時間を解析することは重要である。図６はイベントモニタ画面で状態遷移マシン（６０５）について、いつどの状態６１０にあり、いつイベント６１５を受け付け、そのイベントを処理したかを示している。これらの情報を表示するため、各状態遷移マシンは状態遷移をする時に図７に示す状態遷移記録データをデータベースに格納し、またイベントの発生時およびイベントの処理実行時に図８に示すイベント記録データを作成かつ格納することにする。

【００２３】図６のイベントモニタ画面では、指定された状態マシン６０５について、指定された時間における状態の移り変わりを表示しており、図６の例ではｆｓｍ１は初期状態としてＳ１１から始まり、途中でイベントｅ１１を受け付けたが状態Ｓ11およびＳ１２でイベントｅ１１による状態遷移が定義されていないためにイベントバッファに格納されたままになっている。一方、状態マシンｆｓｍ２は状態Ｓ21のときにイベントｅ２１を読み込んで一旦はイベントバッファに格納しているが状態Ｓ２２でイベントｅ２１による状態遷移が定義されているので個々で処理され、状態Ｓ２３への状態遷移を起こしている。

【００２４】本実施例では状態モデルとしてMealyモデルについて述べたがMooreモデルについても同様の手法を適用することができる。

6

【００２５】

【発明の効果】本発明によれば、状態数の多い複雑なシステムにおいて状態遷移モデルの設計時に状態遷移の定義洩れがあったとしても、入力イベントに対して誤った状態遷移および誤ったアクションを定義しなければ、性能の低下はあるにしてもシステムのデッドロックや誤動作を防げるため、システムの信頼性は向上する。また、性能が厳しく要求されないシステムにおいては、全ての状態について起こり得る全てのイベントに対する状態遷移を考える必要がなく、わかる範囲で誤りのない状態遷移表を作成すれば、正常動作するプログラムが作成できるのでソフトウェアの生産効率は向上する。

【図面の簡単な説明】

【図１】イベントバッファを用いた状態遷移マシンのフローチャート。

【図２】状態遷移の説明図。

【図３】イベントバッファのデータ構造の説明図。

【図４】イベント記憶を用いた状態遷移マシン生成のためのグラフィカルユーザインターフェース説明図。

【図５】グラフィカルユーザインターフェースを用いた状態遷移マシン生成のフローチャート。

【図６】イベントモニタ画面の説明図。

【図７】状態遷移記録データの説明図。

【図８】イベント記録データの説明図。

【符号の説明】

１０５…イベント待ち、１１０…遷移ありなしの判定、１１５…アクション実行、１２０…状態更新、１２５…イベントバッファからイベントを取り出し、１３０…新しい状態に対して状態遷移を起こすか否かの判定。

【図２】

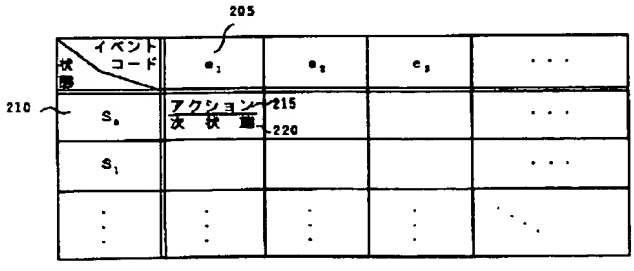
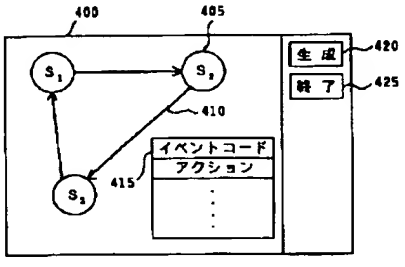


図２

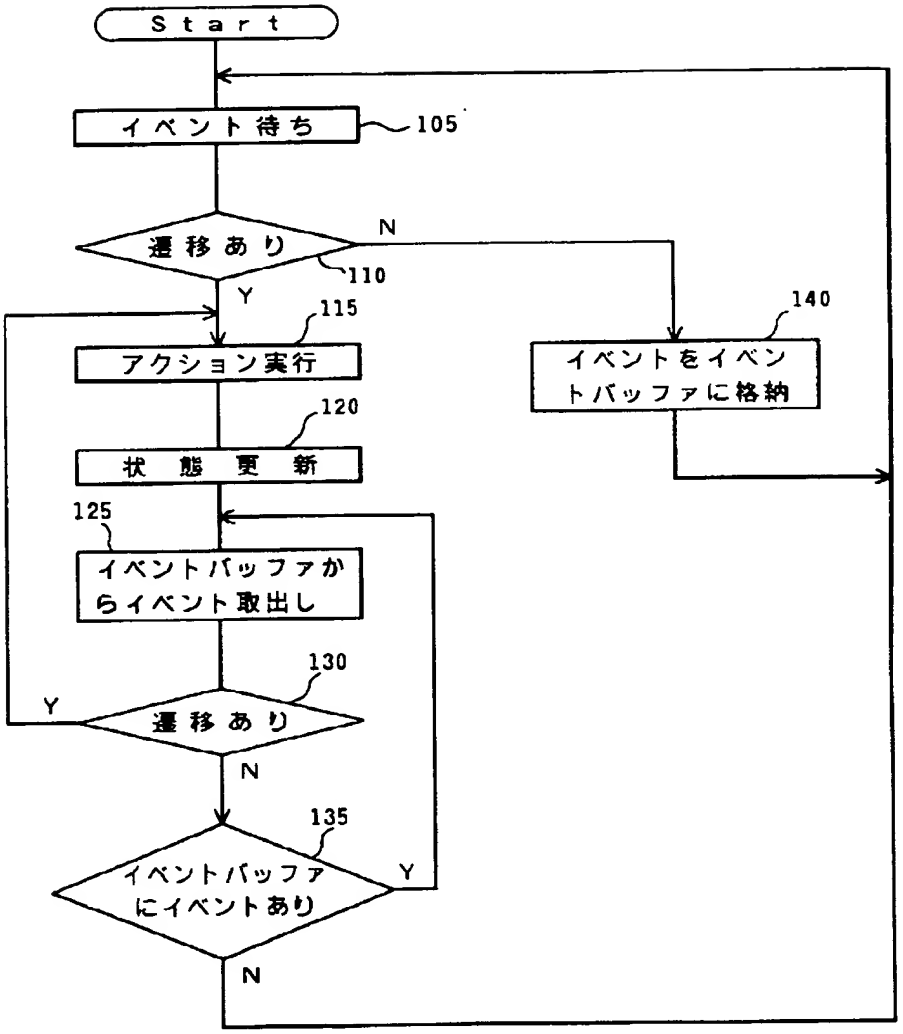
【図４】

図 4

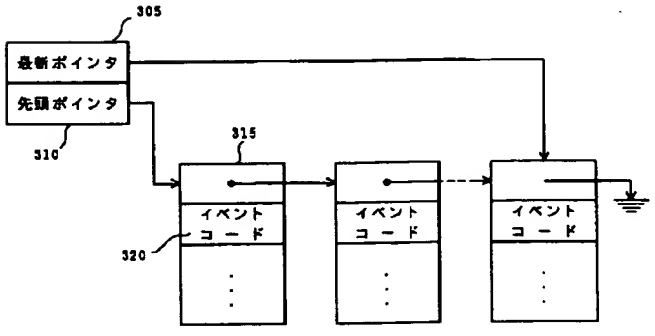


【図１】

図　1

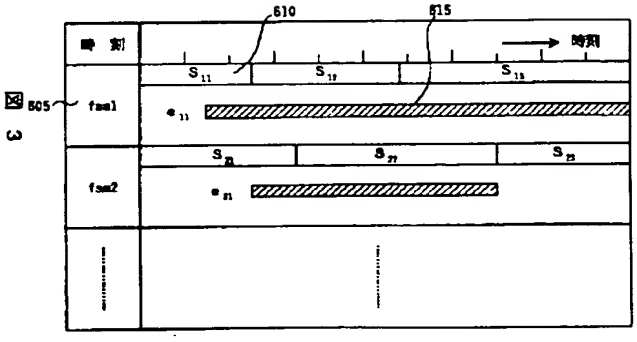

Ｓｔａｒｔ
イベント待ち
105
遷移あり
110
N
Y
115
アクション実行
120
状態更新
125
イベントバッファからイベント取出し
130
遷移あり
Y
N
135
イベントバッファにイベントあり
Y
N
140
イベントをイベントバッファに格納

【図３】


305
最新ポインタ
先頭ポインタ
310
315
イベントコード
320

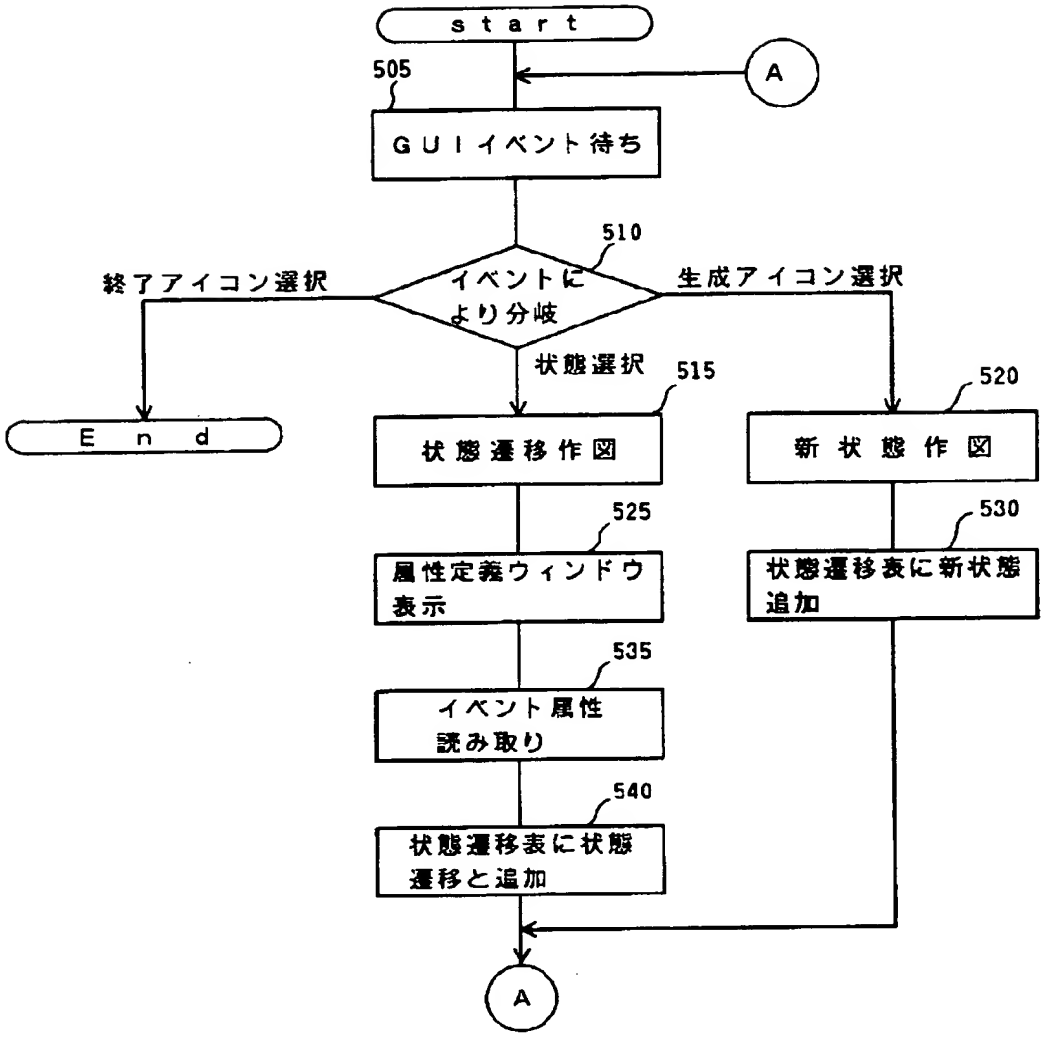
【図６】

図　6


610
615
時刻
時間
fsm1
fsm2
605
図６

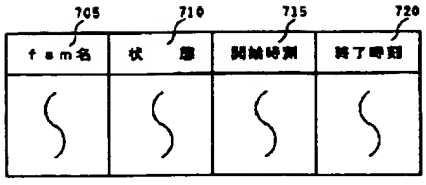
【図５】

図　５



【図７】

図　７



【図８】

図　８